[24] T.H. Jukes and C.R. Cantor, *Evolution of protein molecules* in Mammalian Protein Metabolism, (H.N Munro, ed.), Academic Press, New York, pp. 21-132, 1969.

[25] S. Kannan, E. Lawler, and T. Warnow, *Determining the evolutionary tree*, Proc. First Annual ACM-SIAM Symp. on Discrete Algorithms, San Francisco, Jan. 1990.

[26] S. Kannan and T. Warnow, *Using Experiments to Infer Evolutionary Trees*, manuscript.

[27] M. Křivánek, *The complexity of ultrametric partitions on graphs*, Info. Proc. Letters, Vol. **27** No. 5, (1988) 265–270.

[28] W.-H. Li, *Simple method for constructing phylogenetic trees from distance matrices.*, Proc. Natl. Acad. of Sciences, USA, 78:1085-89.

[29] C. Lund and M. Yannakakis, *On the hardness of approximating minimization problems*, 1992, manuscript.

[30] W. Miller and E.W. Myers, *Sequence Comparison with Concave Weighting Functions*, Bull. Math. Bio. Vol. 50 (2) (1988) 97-120.

[31] N. Saitou and M. Nei, *The neighbor-joining method: a new method for reconstructing phylogenetic trees*, Mol. Biol. Evol. 4:406-25, 1987.

[32] D. Sankoff, *Minimum mutation trees of sequences,* SIAM J. of Applied Math, 28 (1975) pp. 35-42.

[33] R. Sokal and P. Sneath, *Numerical Taxonomy,*, San Francisco: Freeman, 359. pp., 1963.

[34] M.A. Steel, *The complexity of reconstructing trees from qualitative characters and subtrees,* Journal of Classification, Vol. 9, 1992.

[35] E. Sweedyk and T. Warnow, *The Optimal Tree Alignment Problem is Hard*, manuscript, 1992.

[36] R.E. Tarjan, *Sensitivity Analysis of Minimum Spanning Trees and Shortest Path Trees*, Information Processing Letters 14(1):30-33, 1982.

[37] Y. Tateno, M. Nei, F. Tajima, *Accuracy of estimated phylogenetic trees from molecular data. I: Distantly related trees*, J. Mol. Evol., 18:387-404.

[38] M.S. Waterman, T.F. Smith, M. Singh, and W.A. Beyer, *Additive evolutionary trees*, J. Theor. Biol., 64, pp. 199-213, 1977.

[39] W.J. Wilbur, *On the PAM matrix model of protein evolution*, Mol. Biol. Evol. 2:434–447, 1985.

[7] T. Cormen, C Leiserson, and R. Rivest, *Introduction to Algorithms*, MIT Press, 1990.

[8] J. Culberson and P. Rudnicki, *A fast algorithm for constructing trees from distance matrices*, Information Processing Letters, 30 (1989), pp. 215-220.

[9] W.H.E. Day, *Computational complexity of inferring phylogenies from dissimilarity matrices*, Bulletin of Mathematical Biology, Vol. 49, No. 4, pp. 461-467, 1987.

[10] M. Dayhoff and R. Eck, *Atlas of Protein Sequence and Structure 1967-68*, pp. 307. Silver Spring, Maryland: Natl. Biomed. Res. Found.

[11] *Trees and Hierarchical Structures*, Lecture Notes in Biomathematics, ed. Dress and Von Haessler, Springer-Verlag, 1987.

[12] J.S. Farris, *Estimating phylogenetic trees from distance matrices*, Am. Nat., 106, pp. 645-668, 1972.

[13] J. Felsenstein, *Numerical methods for inferring evolutionary trees,* The Quarterly Review of Biology, Vol. 57, No. 4, Dec. 1982.

[14] J. Felsenstein, *Phylogenies from molecular sequences: inference and reliability*, Annual Rev. Genet. 1988, 22:521-65.

[15] W. Fitch, *A Non-sequential method for constructing trees and hierarchical classifications*, J. of Molecular Evolution, 1981.

[16] W.M. Fitch and E. Margoliash, *The construction of phylogenetic trees*, Science 155:29-94, 1976.

[17] M.R. Garey and D.S. Johnson, *Computers and Intractability*, 1979, New York: W.H. Freeman and Company, 340 pp., 1979.

[18] G.H. Gonnet, M.A. Cohen, and S.A. Benner, *Exhaustive matching of the entire protein sequence database*, Science 256:1443–1445, 1992.

[19] D. Harel and R. Tarjan, *Fast Algorithm for Finding Nearest Common Ancestors*, SIAM J. Comput. 13(2):338-355, 1984.

[20] J. Hein, *An optimal algorithm to reconstruct trees from additive distance matrices*, Bulletin of Mathematical Biology, Vol. 51, No. 5, pp. 597-603, 1989.

[21] J. Hein, *A new method that simultaneously aligns and reconstructs ancestral sequences for any number of homologous sequences, when the phylogeny is given*, Mol. Biol. Evol. 6(6):649-668m 1989.

[22] J. Hein, *A tree reconstruction method that is economical in the number of pairwise comparisons used*, Mol. Biol. Evol. 6(6), pp. 669-684, 1989.

[23] C.J. Jardine, N. Jardine, and R. Sibson, *The Structure and Construction of Taxonomic Hierarchies*, Mathematical Biosciences, 1, 173-179, 1967.

We examined the methods by which biologists actually determine the interspecies distances, and discovered that these methods are problematic in that they rely upon criteria which are in themselves *NP-hard*, and also widely debated by the scientific community. We therefore suggested that, rather than restricting our models to exact distances, we would presume that each interspecies distance is given as an interval in which the tree-distance must lie.

Problems based upon constructing additive trees from these generalized distances were generally found to be *NP-complete*, but we were able to find polynomial time algorithms for the problem of constructing ultrametric trees from generalized distances. Since ultrametric trees are what one obtains when the distance is given in terms of time and the species are all current day species, this is a useful tool.

10 Open Problems

One important open question is that of finding a minimum increment to an additive tree under the L^∞ norm. At present, we do not know if there is a polynomial time solution to this problem. Another interesting area of research is in modeling evolution by trees. We know that when we seek additive trees to fit distances, the problems that result turn out to be NP-complete most of the time. On the other hand, ultrametric trees are perhaps unreasonably restrictive since they are premised on a constant rate of molecular evolution. It would be interesting to find intermediate models that model (bounded) variations in rates of evolution where the problems remain tractable. Finally, the question of finding approximately optimal solutions under the L^1 and L^2 norms for additive and for ultrametric trees remains open.

References

[1] A.V. Aho, Y. Sagiv, T.G. Szymanski, and J.D. Ullman, *Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions*, SIAM J. of Computing, vol 10, no. 3, 1981.

[2] S.F. Altschul, *Amino acid substitution matrices from an information theoretic perspective.*, J. Mol. Biol. 219: 555–565, 1991.

[3] W. Beyer, M. Stein, T. Smith and S. Ulam, *A molecular sequence metric and evolutionary trees*, Math. Biosci., 19:9-25.

[4] H. Bodlaender, M. Fellows, and T. Warnow, *Two strikes against perfect phylogeny*, appeared, Proceedings, International Congress on Automata and Language Processing (ICALP), Vienna, Austria, July 1992.

[5] P. Buneman, *Mathematics in the Archeological and Historical Sciences*, (F.R. Hobson, D. G. Kendall and P. Tautu, eds) p. 387, Edinburgh: University Press.

[6] L. Cavalli-Sforza and A. Edwards, *Phylogenetic analysis models and estimation procedures,* Evolution, 32:550-70, 1967.

We will show that Graph Colorability reduces to the Minimum Size Tree problem. We now describe the reduction.

Let $(G = (V, E), k)$ be an instance of graph coloring, with $V = \{v_1, v_2, \ldots, v_n\}$. We will determine an n-by-n matrix $M(G)$ as follows. If $(v_i, v_j) \in E$ we set $M_{ij} = 2$, else we set $M(G)_{ij} = 0$.

We now prove the following lemma.

Lemma 12 *G can be k-colored if and only if there exists an edge-weighted ultrametric tree T such that $d_{ij}^T \geq M(G)_{ij}$ for all i, j, and such that $C(T, w) \leq k$.*

Proof: Suppose G can be k-colored. We construct a tree T as follows. Let r be a root, and let $x_1, x_2, \ldots, x_k$ be the children of r. The children of x_i will be the vertices of the i^{th} color class in G. We then assign weights for each edge by setting $w(v, x_i) = 0$ for all v which are children of x_i. We set $w(x_i, r) = 1$. This tree satisfies the constraints above.

For the converse, suppose such a tree T exists. If T is minimal it can be shown that the maximum interleaf distance in the tree is equal to the maximum entry in the matrix $M(G)$. We can therefore assume that all distances d_{ij}^T are either 2 or 0. Thus, all the children of the root r will be at distance 1 from the root, and the children of these nodes will be at distance 0 from their parents (and hence from each other). Thus, $C(T, w) = |\{x : x \text{ is a child of } r\}| \leq k$. Let the children of r be $x_1, x_2, \ldots, x_q$ for $q \leq k$. We define the coloring $c : V \rightarrow Z$ by setting $c(v) = i$ where v is a child of x_i. To see that this coloring is proper, suppose that $c(v_i) = c(v_j)$ for adjacent vertices v_i, v_j. Then $M(G)_{ij} = 2$, so that in the tree T, $d_{ij}^T = 2$. But then v_i and v_j are not children of the same child of r, so they are colored differently. ∎

We thus have the following theorem.

Theorem 10 *Finding the Minimum Ultrametric Tree is* NP*-complete.*

Because of the linearity in our construction and a theorem of Lund and Yannakakis[29], we can also prove the following:

Theorem 11 *There is an $\epsilon > 0$ such that the Minimum Ultrametric Tree Problem can not be approximated in polynomial time within ratio n^ϵ unless $P = NP$.*

This is, we believe, the first non-approximability result relating to phylogenetic tree construction to appear in the literature.

9 Summary and Conclusions

Computing phylogenetic trees from distances is a standard problem for biologists, despite the fact that distance matrices rarely fit trees exactly, and when they do not, finding an optimal tree (under any of several optimization criteria) is an *NP-hard* problem. Biologists have therefore resorted to using clustering methods (heuristics such as the *UPGMA* method[33]) which do not promise to produce trees of guaranteed error bounds) or to performing exhaustive search, in their search for likely candidates for trees.

ultrametric tree U, $d^U_{ij} \leq 2$, $\forall\, i,j$. In the instance of MIA that we create, the specified distances between the x_i's and the value of k will be unchanged. We introduce $m = n^4$ dummy nodes $y_1, y_2, \ldots y_m$ and specify other distances as follows. For $i \in \{1,2,\ldots,n\}$ and $j,k \in \{1,2,\ldots,m\}$ we specify $M(x_i,y_j) = 4$ and $M(y_j,y_k) = 1$ Then the theorems in [9] serve to establish the following facts about the optimal additive tree, T, whose interleaf distances are an increment to the specified distances.

1. For all $i,j \in \{1,2,\ldots,m\}$, $d^T_{y_iy_j} = 1$ and there is an internal node r in T such that $d^T_{ry_i} = 1/2$.
2. For all $i \in \{1,2,\ldots,n\}$ and $j \in \{1,2,\ldots,m\}$, $d^T_{x_iy_j} = 4$ and $d^T_{x_ir} = 3.5$.

The two points above establish that the optimal *additive tree* on the expanded set of nodes is obtained by (1) finding the optimal *ultrametric tree* on the original set of nodes, (2) appending the internal node r to be at distance 3.5 from each of the original nodes, and, finally, (3) appending each of the y_j's in a separate branch of length 1/2 from r. Thus if we can decide the version of MIA produced above, we can decide the original instance of MIU, proving that MIU is NP-Complete. ∎

8 Approximating the Minimum Size Ultrametric Tree is hard

We consider a different optimization problem in this section. Here we are given a distance matrix M, and we presume that M is a lower bound on the true evolutionary distances, so that the phylogenetic tree T we construct should have $d^T_{ij} \geq M_{ij}$ for all i,j. Within this set of trees we wish to choose an edge-weighted tree T of *minimum size,* where the size of the tree T is $C(T,w) = \Sigma_{e \in E(T)} w(e)$, where $w(e)$ is the weight of the edge e. This criterion was first proposed by Beyer et. al. in 1974 in [3] and Waterman et. al. in 1977 in [38], and is one of the standard criteria used for tree reconstruction. We will show that a special case of this problem is equivalent to computing Graph Coloring, and thus is difficult in two ways: first, it is *NP-complete*, and secondly, unless *P=NP*, there is an $\epsilon > 0$ so that it cannot be approximated within a ratio of n^ϵ.

8.1 Computing the Minimum Size Ultrametric Tree is *NP*-hard

The statement of our problem is as follows:

Minimum Size Ultrametric Tree (MUT)

Input: An n-by-n distance matrix M, positive integer k.

Question: Does there exist an edge-weighted tree T such that $d^T_{ij} \geq M_{ij}$ for all i,j and such that $\Sigma_{e \in E(T)} w(e) \leq k$?

Suppose the answer to SCP was "yes" and that G' is a subgraph of G created by removing at most k edges, such that G' is the disjoint union of cliques. We will create a matrix M' as follows. If $(v_i, v_j) \in E(G')$ then set $M_{i,j} = M'_{i,j} = 1$, else we set $M'_{i,j} = 2$. We claim that the resulting matrix is ultrametric. To see this, suppose a, b, and c are three of the vertices and suppose it is not true that $M'_{a,b} \leq \max(M'_{b,c}, M'_{a,c})$. Then since all distances are 1 or 2 it must be that $M'_{a,b} = 2$ and $M'_{b,c} = M'_{a,c} = 1$. But this means that (b, c) and (a, c) are in $E(G')$, but that $(a, b) \notin E(G')$, contradicting the fact that G' is a disjoint union of cliques. It should also be noted that the net increment from M to M' is within the bound specified. Thus, we have a "yes" instance of MIU.

Conversely suppose that we created a "yes" instance of MIU. It can be shown that the ultrametric matrix we get by incrementing the given matrix, M, will still only contain the entries 1 and 2. Under this assumption suppose M' is the matrix we get after incrementation. Consider creating G' from G by deleting the edges in G corresponding to entries that went from 1 in M to 2 in M'. We claim that G' is a disjoint union of cliques. To prove this we only need to show that if (a, b) and (b, c) are edges in G', then (a, c) is an edge as well. But this follows from the fact that $M'(a, c) \leq \max(M'(a, b), M'(b, c))$ and the fact that $M'(a, b) = M'(b, c) = 1$. These two facts imply that $M'(a, c) = 1$ which implies that (a, c) must be an edge in G'. ∎

7 Minimum Increment to Additive

We now show how we can use the fact that MIU is NP-complete to show that under the L^1-norm a decision version of the *minimum increment to additive* problem is also *NP-hard.* Our proof of this reduction is essentially identical to a reduction given by Day[9]. Day[9] shows that under the L^1-norm, the problem of finding the best-fit ultrametric tree reduces to the problem of finding the best-fit additive tree. There are two differences between the reduction that [9] presents and the reduction that we seek. First, [9] is concerned with best-fit, while we are concerned with min. increment, and second, [9] allows only half-integer weights on the tree edges, while we allow arbitrary (finite precision) real weights. We define the problem MIA as follows.

L^1-Minimum Increment to Additive Problem (MIA)

Input: A non-negative distance matrix M and a real number k.

Question: Is there an edge-weighted additive tree T such that $d^T_{ij} \geq M_{ij}\ \forall i, j$ and $\sum_{ij}(D^T_{ij} - M_{ij}) \leq k$?

Theorem 9 *The L^1-Minimum Increment to Additive Problem is* NP-Complete.

Proof: We have shown that the minimum increment problem to ultrametric (MIU) in NP-complete even in the special case where the entries in the matrix take on only the values 1 and 2. We will make the reduction to MIA from this special case of MIU. Suppose the input to the special case of MIU is an $n \times n$ matrix M and an integer k. Denote the n-element species set implicit in the MIU problem by $x_1, x_2, \ldots, x_n$. As noted before, in any optimal

triangles in G. Specify k to be $n^3/3 + (T - n/3)3n^2 + m - n$. We claim that the instance of SCP is a 'yes' instance iff the instance of PiT that we started with was a 'yes' instance.

Suppose we started with a 'yes' instance of PiT. Then there is a set S of $n/3$ triangles which partition the vertices of G. In H we retain the following cliques: For each $t \in S$ we retain the clique on $n^2 + 3$ nodes formed by the vertices of $t \cup K(t)$. For each $t \notin S$ we retain the $n^2 + 1$-sized clique formed by the vertices of $K(t) \cup s(t)$. We remove any edges that are not part of these cliques. We count the number of edges removed as follows: For each $t \in S$ we need to remove the n^2 edges that connect $K(t)$ to $s(t)$. Since $|S| = n/3$ this amounts to a total of $n^3/3$ edges. For each $t \notin S$ we remove the $3n^2$ edges connecting t with $K(t)$. This amounts to a total of $(T - n/3)3n^2$ edges. Finally, n of the edges in G are covered by the triangles used in S and hence by the clique cover we have defined above. We need to remove the remaining $m - n$ edges. The total number of edges removed is exactly equal to the bound k we have above and hence we have a 'yes' instance of SCP.

Conversely suppose we have a 'yes' instance of SCP. Let t be any triangle in G. Let us focus on the edges between $K(t)$ and the rest of the graph. Each vertex in $K(t)$ has three edges to t and an edge to $s(t)$. Of these four edges, one, two, or three edges will be covered by cliques in an optimal partition into cliques. If three edges are chosen then the optimal partition must include the clique on $t \cup K(t)$. Let r be the the number of these $K(t)$ cliques covered by $(n^2 + 3)$-sized cliques chosen in the partition, and l be the number of these cliques covered by $(n^2 + 2)$-sized cliques in H obtained by choosing two of the vertices in G out of the triangle to which each of these cliques is attached.

Note that $3r + 2l \leq n$ and that the total number of edges between the vertices of cliques $K(t)$ and the rest of H that is not covered by cliques in the clique partition is at least $(r + 2l + 3(T - r - l))n^2$. Call such edges *cross-edges*. We will show that the number of these cross-edges which are not covered by an optimal clique partition exceeds our bound if G does not have a partition into triangles. Assume that G does not have a partition into triangles. Then since $r < n/3$, the minimum number of these cross-edges that are left uncovered is $(n/3 - 1)n^2 + 2n^2 + 3(T - n/3)n^2$. This corresponds to the choice of $r = n/3 - 1$ and $l = 1$. This is already greater than the bound on the number of edges that can be removed since $n^2 > m - n$. Thus we could not have had a 'yes' instance of SCP. This establishes the validity of the reduction. ∎

6.2 Minimum Increment to Ultrametric Problem is *NP-complete*

We know from the previous section that the Subgraph Clique Partition Problem (SCP) is *NP-complete*. We will prove the Minimum Increment to Ultrametric Problem (MIU) is *NP-complete* by reducing SCP to MIU.

Theorem 8 *The Minimum Increment to Ultrametric Problem is* NP-complete.

Proof: Given an instance (G, k) of SCP, we create a distance matrix as follows: If $(v_i, v_j) \in E(G)$ we set $M_{ij} = 1$, else we set $M_{ij} = 2$. We then ask if the matrix M can be incremented to an ultrametric matrix M' by incrementing the entries by no more than a total of k. We claim that the answer to this question is "yes" iff the answer to the instance of SCP was "yes".

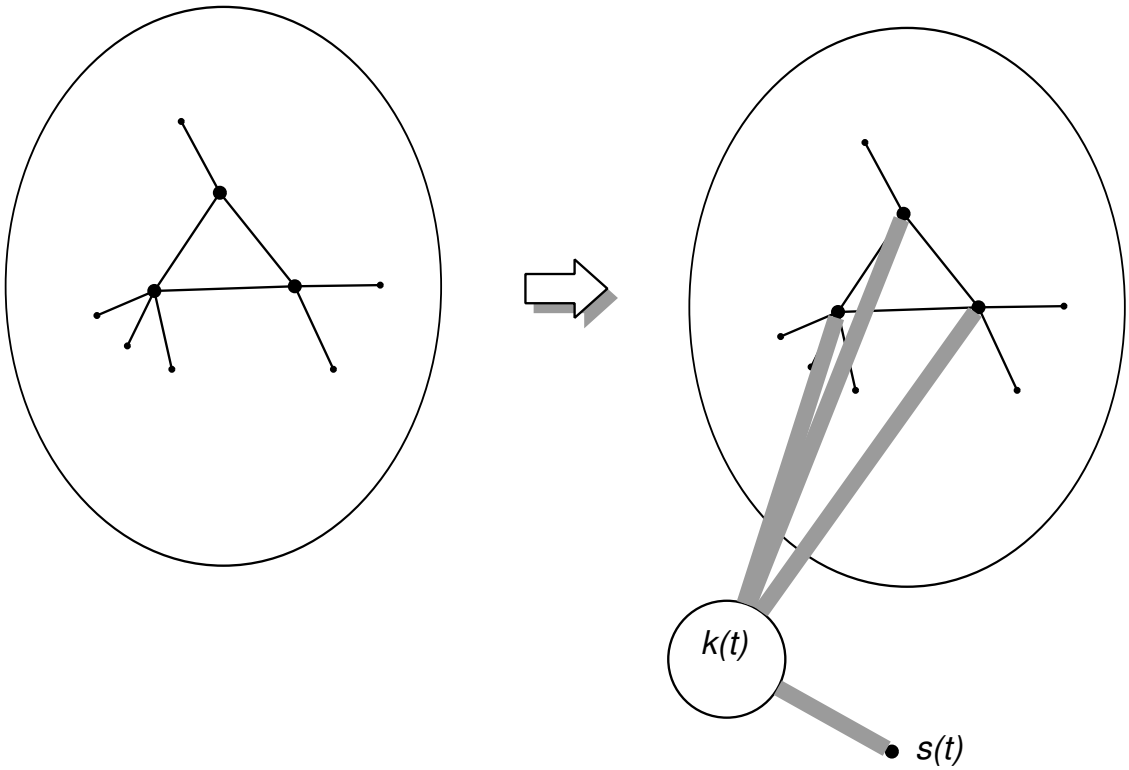

Figure 4: Transformation of PiT to SCP

Partition into Triangles (PiT) Problem

Input: Graph $G = (V, E)$ with $|V| = 3q$ for some integer q.

Question: Can the vertices of G be partitioned into q disjoint sets $V_1, V_2, \ldots, V_q$ so that each V_i induces a triangle?

The PiT Problem is listed as *NP-Complete* in [17]. We can now prove that SCP is *NP-complete.*

Theorem 7 *The Subgraph Clique Partition Problem is* NP-complete.

Proof:

We show that PiT reduces to SCP.

Let G be an instance of PiT. We construct the following graph H as an instance of SCP. We also need to specify the integer k as part of the input to SCP. Figure 4 illustrates the transformation.

Let n be the number of vertices in G. For every triangle $t \in G$ attach a clique $K(t)$ on n^2 new vertices to the three vertices in the triangle creating a clique on $n^2 + 3$ vertices. To each clique $K(t)$ on n^2 new vertices that we added, attach one more new vertex, $s(t)$. Thus each $K(t)$ connects on one side to a single vertex $s(t)$ and on the other side to a triangle t. This is the graph H. Let m the number of edges in G and T the number of

5.2 Matrix Sandwich to Additive Problem

Since the Matrix Completion to Additive is *NP-complete* the Matrix Sandwich to Additive is also trivially *NP-complete.*

Lemma 11 *The Matrix Sandwich to Additive is* NP-complete.

Proof: We will reduce MCA to MSA. Let M be the input to MCA, and let E be the set of pairs (i,j) such that $M[i,j]$ is set. For each $(i,j) \in E$, set $M_h[i,j] = M_l[i,j] = M[i,j]$, and for each $(i,j) \notin E$, let $M_l[i,j] = 0$ and $M_h[i,j] = \infty$. It is clear that any solution to the MSA problem yields a solution to the MCA problem, and vice-versa. We showed in the previous section that MCA is *NP-complete*, so that MSA is also *NP-complete.* ∎

6 Hardness of min increment to ultrametric under L^1-norm

In this section we show that the problem of minimally incrementing a distance matrix under the L^1-norm so as to make it an ultrametric matrix is *NP-hard.* We define an equivalent decision problem, which we call the *Minimum Increment to Ultrametric* Problem, or MIU.

L^1-Minimum Increment to Ultrametric Problem (MIU)

Input: A non-negative distance matrix M and a real number k.

Question: Is there an edge-weighted tree T such that $d_{ij}^T \geq M_{ij} \forall i,j$ and $\sum_{ij}(D_{ij}^T - M_{ij}) \leq k$?

The NP-completeness proof involves proving an intermediate problem, the *Subgraph Clique Partition Problem*, to be *NP-complete.* We will define the Subgraph Clique Partition (SCP) Problem, and prove SCP *NP-complete* by a reduction from the *NP-complete problem*, Partitioning into Triangles (or PiT). We will then show that SCP reduces to MIU, so that MIU is *NP-complete.*

6.1 The Subgraph Clique Partition Problem is *NP-complete*

We begin by defining the Subgraph Clique Partition Problem.

Subgraph Clique Partition Problem (SCP)

Input : Graph $G = (V,E)$, integer k.

Question: Does there exist a set $E_0 \subset E$ of k or fewer edges so that in the graph $G' = (V, E' = E - E_0)$ if $(u,v) \in E'$ and $(v,w) \in E'$ then $(u,w) \in E'$.

In other words we are asking if k or fewer edges can be removed from G such that all the connected components of the resulting graph are cliques.

To show that SCP is *NP-complete*, we will reduce the *Partition Into Triangles (PiT)* problem to SCP. The PiT problem is as follows:

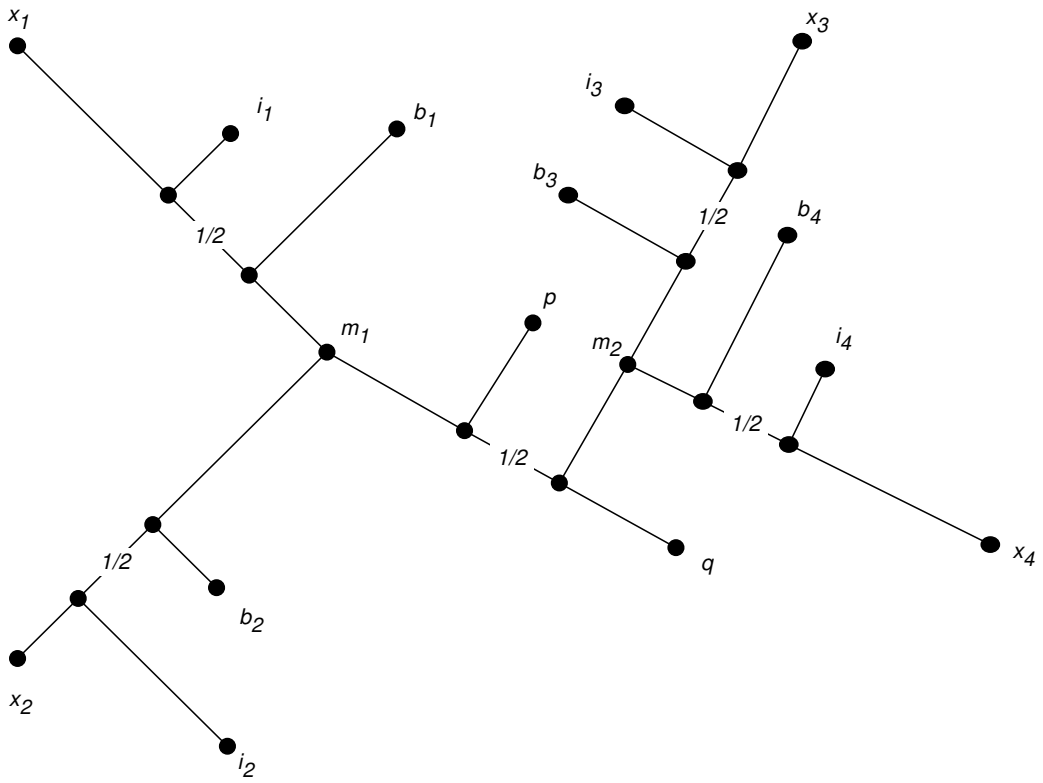

Figure 3: Transformation of a quartet into a set of flexible-distance gadgets

For every quartet we create the collection of gadgets and other distance specifications as above using a completely new set of dummy nodes. We can then argue that there is a tree consistent with the distance constraints iff there was a tree consistent with the quartets that were input to QCP. This will prove that MCA is *NP-hard*.

Suppose there is a tree T that is consistent with the quartets. By Lemma 10, we can assume that no interleaf distance in T is more than $n-1$ times any other. If $\{(i_1, i_2)(i_3, i_4)\}$ is a quartet that is part of the input to QCP, then since T is consistent with all such quartets, there must exist an edge $e = (u, v) \in T$ whose removal separates (i_1, i_2) from (i_3, i_4). To produce a tree that is consistent with the distance constraints imposed by the transformation of the quartet $\{(i_1, i_2)(i_3, i_4)\}$, let m_1 be the vertex where the path in T from i_1 to u first intersects the path from i_2 to u and similarly let m_2 be the vertex where the path from i_3 to v first intersects the path from i_4 to v. Let i_1, i_2, i_3, i_4 be placed in the same positions at which they occur in T. Choose the position of the other dummy nodes in order to produce the correct distances between i_1m_1, i_2m_1, i_3m_2, i_4m_2 and m_1m_2. This shows that there is a tree that satisfies the distance conditions specified. Conversely, note that by property (1) of the flexible-distance gadget, any tree that is consistent with the specified distances is also consistent with all quartets. ∎

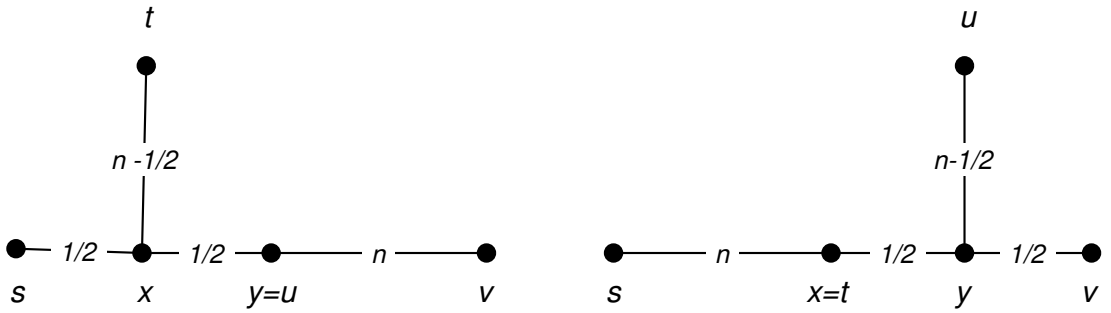

Figure 2: The two extreme arrangements of a flexible-distance gadget

The first point is a simple consequence of the four-point condition stated in Section 2. Note that $d(s,v)+d(t,u) = 2n+1 > 2n = d(s,t)+d(u,v)$. Thus, the only way that these distances can satisfy the four-point condition is if $d(s,u)+d(t,v) = 2n+1$, so that the topology constraint, $\{(s,t),(u,v)\}$, is met.

We now prove the second point. By the topology constraint, in any tree realization of the specified distances, when following the path from s to v we will first encounter the branch containing t and subsequently the branch containing u (Figure 2). Let x be the point along the s to v path at which the branch containing t is attached and let $d(s,x) = k$. Then $d(x,t) = n-k$ and $d(x,u) = n-(n-k) = k$. If y is the point at which the branch containing u attaches to the path from s to v, then y must be equidistant from t and v since u is equidistant from t and v. We have, $d(t,v) = d(s,v) - d(s,x) + d(x,t) = 2n+1-2k$ and hence $d(t,y) = n+1/2-k$. In other words, $d(x,y) = 1/2$. It can be seen that k can take on any value between $1/2$ and n while keeping all distances non-negative and that $d(s,u) = 2k$ ranges between 1 and $2n$. We will denote this construction by $fg(s,t,u,v)$; in this construction, the distance between the first and the third argument is flexible and can vary between 1 and $2n$. (Figure 2 shows how the gadget realizes the two extreme values of $d(s,u)$.)

Theorem 6 *The Matrix Completion to Additive Problem is* NP-complete.

Proof: It is easy to check that the MCA Problem is in *NP*. We now show that it is also *NP-hard*. We will describe how to encode quartets by flexible distance gadgets which impose the correct topology constraints without really imposing serious distance constraints on the tree. Let $\{(i_1,i_2),(i_3,i_4)\}$ be a quartet. We construct the following flexible distance gadgets: $fg(m_1,b_1,i_1,x_1)$, $fg(m_1,b_2,i_2,x_2)$, $fg(m_2,b_3,i_3,x_3)$ and $fg(m_2,b_4,i_4,x_4)$ and also the flexible distance gadget $fg(m_1,p,m_2,q)$. Figure 3 illustrates the gadgets used in the transformation of the quartet $\{(i_1,i_2)(i_3,i_4)\}$.

In addition to the distances specified by the definition of the flexible distance gadgets we specify the following distances: $d(x_1,x_2) = 2n+2$ to enforce that x_1 and x_2 are on opposite sides of m_1 and similarly $d(x_3,x_4) = 2n+2$ to ensure that x_3 and x_4 are on opposite sides of m_2. We set $d(x_1,p) = d(x_2,p) = d(x_3,p) = d(x_4,p) = 2n+1$ to enforce the correct conditions of betweenness.

Input: A set S of pairs of pairs, of the form $\{(i,j),(k,l)\}$.

Output: A leaf labeled tree T (if one exists) such that for each element $\{(i,j),(k,l)\}$ in the set S, there is an edge $e \in T$ such that $T - \{e\}$ has i,j in one subtree and k,l in the other subtree.

Pairs of pairs of the form $\{(i,j),(k,l)\}$ are called *quartets* and they impose a constraint on the topology of the tree by requiring i and j to be on 'one side' and k and l to be on the other. The Quartet Consistency Problem was shown to be *NP-complete* in [34], and is equivalent to the Perfect Phylogeny Problem, which was independently shown to be *NP-complete* in [4]. Both formulations are explicitly concerned with constructing phylogenetic trees from character sets.

We state the following easy lemma for use in our reduction.

Lemma 10 *If there is a tree, T, that is consistent with a given set of quartets on n species, then there is an edge-weighted tree, T', which is also consistent and where the largest interleaf distance is at most $n-1$ times the smallest interleaf distance.*

Proof: We can assume without loss of generality that no internal vertex in the tree has degree two. Now, assign weights of one to all the edges of T, and the proposition follows. ∎

The Reduction

QCP imposes constraints on the topology of the tree without imposing any constraints on the distances between nodes. On the other hand an instance of MCA explicitly specifies some of the distances between nodes. Thus in reducing QCP to MCA we need to have a way of imposing topology constraints among quartets without constraining the distances between the four nodes in the quartet. In order to do this we will introduce *dummy nodes* in addition to the nodes that are part of the input to QCP which we call the *original nodes*. We will never specify the distance between any pair of original nodes. Instead we will impose the topology constraints of QCP by specifying distances between original nodes and dummy nodes and between pairs of dummy nodes. Our constraints do have the side-effect of implicitly forcing distances between pairs of original nodes to lie within a certain range. We then appeal to Lemma 10 and argue that the range is big enough to allow for the realization of a tree consistent with all the quartets if one exists. All this is accomplished using the primitive of a *flexible-distance gadget* which is described below.

Flexible-distance gadget Suppose s,t,u,v are four nodes and the following distances are specified between them: $d(s,t) = n,\ d(t,u) = n,\ d(u,v) = n,\ d(s,v) = n+1$ where n is some positive integer to be specified later. These are the specifications of a flexible-distance gadget on s,t,u, and v and we will prove the following two properties about the gadget.

1. The gadget imposes the topology constraint corresponding to the quartet $\{(s,t),(u,v)\}$.

2. The (unspecified) distance between s and u can attain any value between 1 and $2n$.

This problem includes as special cases, both the min increment and the best-fit problems under the L^∞ metric. To see this, note that setting $f(M[i,j],\epsilon) = M[i,j] - \epsilon$ and $g(M[i,j],\epsilon) = M[i,j] + \epsilon$ defines the best-fit problem, while changing the definition of f to $f(M[i,j],\epsilon) = M[i,j]$ defines the min. increment problem under the L^∞ norm. We now show an $O(e + n\log n)$ algorithm for it derived from a simple modification to our sandwich algorithm.

Theorem 5 *There is an $O(e + n\log n)$ algorithm for the Ultrametric L^∞ optimization problem.*

Proof: Construct the minimum spanning tree T using the weights given by the weight-function M. Compute cut-weights using again the same weight-function M. At this point choose the minimum ϵ to ensure separability. To do this, we need to make sure that for all edges e on the spanning tree with weight $M[e]$, the cut-weight $CW(e) \leq M[e]$. By the monotonicity of f and g it is enough to ensure that ϵ is chosen big enough in order to satisfy $f(CW(e),\epsilon) \leq g(M[e],\epsilon)$. Thus each edge of the spanning tree imposes a lower bound on ϵ and we just take the maximum of these ϵ's (in linear time) as the optimal choice of ϵ. We can then compute the best tree by evaluating f and g explicitly for each of the edge-weights, solving for ϵ, and solving a sandwich problem. ∎

The above theorem can be generalized to optimize for a fixed number of parameters such as ϵ, but the optimization under the L^1 norm where we want to minimize the sum of (up to) n^2 different increments turns out to be *NP-hard.*

5 Results for Additive Trees

By contrast with our results about ultrametric trees, we will show that the Matrix Completion Problem is *NP-complete* for general trees, so that the Matrix Sandwich Problem is also *NP-complete.* We begin with the Matrix Completion Problem.

5.1 The Matrix Completion to Additive Problem is *NP-complete*

Recall the Matrix Completion to Additive Problem.

Matrix Completion to Additive Problem (MCA)

Input: Matrix M for which only some of the entries are known. An unset matrix entry is indicated by $M[i,j] =$ "*".

Question: Does there exist an *additive* matrix M' satisfying $M[i,j] = M'[i,j]$ whenever $M[i,j] \neq$ "*"?

We will prove that the MCA problem is *NP-hard* by reducing *Quartet Consistency* to MCA. The Problem of Quartet Consistency is as follows:

Quartet Consistency Problem (QCP)

have computed the cartesian tree for all the sets in our union-find data structure. When examining edge (u, v) in T, we first find the sets, S_u and S_v containing u and v respectively in the union-find data structure. Let $T_C(u)$ and $T_C(v)$ be the cartesian trees constructed for these sets. We join the roots of $T_C(u)$ and $T_C(v)$ to a new root. We then union S_u and S_v.

Sorting the edges of the T takes time $O(n \log n)$ and building the cartesian tree bottom up takes time $O(n\alpha(n, n))$. The total time is then $O(e + n \log n)$ for finding all cut-weights.

The output ultrametric tree U is also a cartesian tree except that the weights used in the definition of this cartesian tree are the cut-weights. This follows from the fact that if $e_1 \in T$ is the edge with the largest cut-weight, then the pairs of nodes (u, v) for which $d^U_{u,v} = CW(e_1)$ are precisely those pairs for which u lies in one subtree formed by the removal of e_1 from T, and v lies in the other. These are precisely the set of pairs for which the least common ancestor in U is the root. This implies that the algorithm we described above for building a cartesian tree can also be used to build U. The only modification required is to assign weights to the edges of U. We prescribe these edge-weights in U implicitly by defining the height of each subtree that we build. We let $H(r)$ denote the height of the subtree rooted at r. We now spell out the algorithm for building U. We set $H(l) = 0$, for each leaf l. We sort the edges of T in increasing order of cut-weights. We maintain a union-find data structure. When processing the edge $(u, v) \in T$, we find the subtrees containing u and v and make the roots of these subtrees children of a new root r. We set $H(r) = CW(u, v)/2$, to ensure that nodes whose least common ancestor is r are at a distance of $CW(u, v)$ from each other. Since we process the edges of T in increasing order of cut-weights, the height function is monotone non-decreasing and hence no edge of U gets assigned a negative weight. ∎

Corollary 9 *The matrix completion to ultrametric problem can be solved in time* $O(n \log n + e)$.

4.5 Optimizing under a generalized L^∞-norm

In this subsection we consider a generalization of the Min. increment and Best-fit problems for ultrametric trees under the L^∞-norm and present an $O(e + n \log n)$ algorithm for this problem. We define the following problem which generalizes both the min. increment and the best-fit problems for ultrametric trees under the L^∞-norm.

Ultrametric L^∞ optimization

Input: A graph $G = (V, E)$ along with a weight function M on the edges and two functions $f(x, \epsilon)$ and $g(x, \epsilon)$ which take two real arguments such that both f and g are monotone non-decreasing on their first argument, f is monotone non-increasing on ϵ and g is monotone non-decreasing on ϵ.

Output: The smallest value of ϵ such that there exists an ultrametric tree U in which for all (i, j) for which $M[i, j]$ is defined, $f(M[i, j], \epsilon) \leq d^U_{ij} \leq g(M[i, j], \epsilon)$.

The above theorem suggests the following algorithm. In the algorithm, $H(T)$ will refer to the distance from the root to the leaves of an ultrametric tree T.

Algorithm B *Sandwich to Ultrametric - MST*

- Build an MST of G_h and compute cut-weights for all edges in the MST.
- Sort the edges of the MST in decreasing order of their cut-weights as $e_1, \ldots, e_{n-1}$.
- Build the ultrametric tree as follows: Let the root node of the tree have two children. Build the left and right subtrees by recursing on the two components created by the removal of e_1. Let T_1 and T_2 be the recursively constructed subtrees. Attach the root to subtree T_i, $i = 1, 2$ with an edge of weight $CW(e_1)/2 - H(T_i)$ and set $H(T) = CW(e_1)/2$ for the resulting tree T.

The correctness of the algorithm above follows from Theorem 3 and by noting that the weights assigned to edges in the ultrametric tree constructed, realize the inter-leaf distances specified by the theorem. Before undertaking an analysis of the running time, we need to look more carefully at the data structures involved.

4.4 Implementation Details and Running Time Analysis

Theorem 4 *The ultrametric graph sandwich problem can be solved in $O(e + n \log n)$ time.*

Proof: It is known that the MST of any graph can be computed in time $O(e+n \log n)$ (see e.g. [7]). Computing cut-weights for the edges of the MST can also be done in $O(e+n \log n)$ as follows.

Let a *Cartesian Tree* of a weighted tree T be a tree T_C such that the root r of T_C is the largest weight edge, e_{max}, in T, and the two children of r in T_C are the recursively defined cartesian trees of the subtrees induced in T by removing e_{max}. The leaves of T_C are the nodes of T.

If we can build a cartesian tree T_C of the MST T using the weights defined by the weight function M_h, then we can find the cut weights as follows. We preprocess T_C for least common ancestor (`lca`) queries. This can be done in linear time so that given any pair of nodes of T_C, we can find their `lca` in constant time [19]. Now for each edge $(u, v) \in E_l$, let l_u, l_v be the leaves in T_C representing nodes u and v. In $O(1)$ time we can find the largest weight edge in $P_T(u, v)$ by finding $\texttt{lca}(l_u, l_v)$ in T_C. So in time $O(e)$ we can find the link-edge for every pair $(x, y) \in E_l$. Cut-weights for every edge in T can also be computed in the same time bounds. In the above algorithm for computing link-edges, we simply add the following computation: For each edge $e \in T$ we initialize its cut-weight to 0 at the beginning of the algorithm, and whenever e gets assigned as the link-edge for a pair (u, v), we update the cut-weight of e to $M_l[u, v]$ if $M_l[u, v]$ exceeds the current cut-weight of e.

To complete the construction, we show how to build a cartesian tree in time $O(n \log n)$. We examine the edges of tree T in increasing order of weights. We maintain a union-find data structure on the leaves of the tree we are building. Suppose inductively that we

Thus, x and y are separable iff $M_l[x,y] \leq D(T,x,y)$.

A *link edge*, $LINK(a,b)$, is an edge in $P_T(a,b)$ with weight $M_h(LINK(a,b)) = D(T,a,b)$. (cf. *sensitivity analysis à la* Tarjan [36]).

For each edge e in T,

cut-weight $CW(e) = \max\{M_l(x,y) | LINK(x,y) = e\}$.

Theorem 3 *There exists an ultrametric tree $U \in [M_l, M_h]$ iff each pair of vertices is separable.*

Proof: Suppose there is a pair (x,y) that is not separable. This means that $M_l[x,y] > M_h[LINK(x,y)]$. If e is an edge in the path $P_T(x,y)$, then $M_h[e] \leq M_h[LINK(x,y)]$. Thus in the cycle created by $P_T(x,y)$ and the edge (x,y), any assignment of distances to the edges of the cycle that respects the sandwich constraints will force the edge (x,y) to be the unique one of maximal weight. This is inconsistent with an ultrametric metric.

Conversely suppose that every pair is separable. Sort the edges in the spanning tree in decreasing order of their cut-weights. Let the edges of the MST be $e_1, \ldots e_{n-1}$ in this order. Consider removing the e_i from the tree successively in this order. For every pair (a,b) which is separated for the first time by the removal of e_i, set the distance, d^U_{ab}, between a and b equal to the cut-weight of e_i. In other words, d^U_{ab} is set equal to $CW(e_i)$ iff e_i is the first edge in $P_T(a,b)$ which is removed. We claim that the resulting set of distances are ultrametric and that they satisfy the constraints imposed by M_l and M_h.

Once again we use the characterization that a set of distances is ultrametric iff in every cycle the maximum distance is attained by more than one edge. Consider the edge-weighted graph that corresponds to the set of distances defined above. Let γ be a cycle and let $e = (u,v)$ be a maximum weight edge in γ. Clearly the weight of e was set equal to the cut-weight of the link edge whose removal separated u and v for the first time. Note also that since e is of maximal weight in C, no other edge of γ could have had its endpoints separated before e. However since every cycle intersects every cut an even number of times, there must be another edge $e' = (u',v')$ in γ whose end-points where also separated for the first time by the removal of that link edge and hence $d^U_{uv} = d^U_{u'v'}$.

We now have to check that the distances we set are within the constraints imposed by the upper and lower bounds. Let (u,v) be an arbitrary pair of nodes and e be the edge whose removal separated (u,v) for the first time. We know that the cut-weight of $LINK(u,v)$ is at least $M_l[u,v]$. Either $e = LINK(u,v)$ or e has cut-weight greater than or equal to $LINK(u,v)$ since e was the first edge removed on $P_T(u,v)$. In either case, it follows that $CW(e) \geq M_l[u,v]$.

To check the upper bound, from the definition of cut-weights, it follows that there is a pair of nodes (p,q) such that $LINK(p,q) = e$ and $M_l[p,q] = CW(e)$. Applying separability to the pair (p,q), it follows that $CW(e) = M_l[p,q] \leq M_h(e)$. Furthermore, since T is a minimum spanning tree, we must have $M_h(e) \leq M_h[u,v]$ since otherwise, e would have been replaced by (u,v) in T. Putting these two inequalities together we get $CW(e) \leq M_h[u,v]$, proving that the distance assigned to (u,v) is within the bounds specified. ∎

Proof: The correctness follows from Lemma 6 and Corollary 5. The running time follows from the fact that finding the maximum distance, finding the connected components and splitting them can all be accomplished in $O(e)$. The algorithm can recurse for up to n levels since at each stage, the size of the maximal component is reduced by at least one. The total time is therefore $O(en)$. ∎

4.3 An $O(e + n \log n)$ Algorithm for Sandwich to Ultrametric

In the brute force algorithm, we create a graph $G_{a,b} = (V, E_{ab}, W)$ in which (x, y) is an edge if $M_h[x, y] < M_l[a, b]$. We take $O(e)$ time to create such a graph and we are interested in the connected components containing a and b. We now show how to obtain this information for all pairs (a, b) using a single minimum spanning tree computation!

We begin by proving some general properties of graphs. Throughout this section, when we refer to a path, we will mean a *simple* path. Let $G = (V, E, W)$ be any weighted graph. For any pair $x, y \in E$, let $P = [x = v_1, \ldots, v_k = y]$ be a path from x to y. We define the cost $c(P)$ of path P to be the weight of the maximum weight edge in P, i.e. $c(P) = \max\{W(v_i, v_{i+1}) | 1 \leq i \leq k-1\}$. Let $\mathcal{P}_G(x, y)$ be the set of paths from x to y. We define $D(G, x, y) = \min\{c(P) | P \in \mathcal{P}_G(x, y)\}$.

For the pair G_l, G_h, let $G_{x,y} = (V, E_{x,y})$, with $E_{x,y} = \{(a, b) | M_h[a, b] < M_l[x, y]\}$. The following lemma establishes a structural property of the graphs $G_{x,y}$.

Lemma 7 *The nodes x and y are in the same component of $G_{x,y}$ iff $M_l[x, y] > D(G_h, x, y)$.*

Proof:

If $M_l[x, y] > D(G_h, x, y)$, then there is a path P from x to y in G_h such that for each edge $e \in P$, $M_h(e) \leq D(G_h, x, y) < M_l[x, y]$. P will be a path in $G_{x,y}$ as well. Conversely if x and y are in the same component of $G_{x,y}$, then there is a path between them in $G_{x,y}$. From the definitions of $G_{x,y}$ and $D(G_h, x, y)$, it follows that $D(G_h, x, y) < M_l[x, y]$. ∎

Now let T be any minimum spanning tree of G_h. We let $P_T(x, y)$ be the path from x to y in T. (Since we are only dealing with simple paths, $P_T(x, y)$ is unique.) We define the distance function $D(T, x, y) = c(P_T(x, y))$.

Lemma 8 $\forall a, b \in V,\ D(T, a, b) = D(G_h, a, b)$.

Proof: Since a path in T is a path in G_h, clearly $D(G_h, a, b) \leq D(T, a, b)$. Now suppose $D(G_h, a, b) < D(T, a, b)$. Then there exists a path P in G_h from a to b such that $c(P) < D(T, a, b)$. Let (c, d) be a maximal weight edge in $P_T(a, b)$. Removing (c, d) from T disconnects a from b. Let (e, f) be the first edge in P whose addition to $T - (c, d)$ would reconnect the tree. Now $W(e, f) < W(c, d)$ so $T \cup (e, f) - (c, d)$ is a tree with total weight less than T, violating the minimality of T. So $D(G_h, a, b) = D(T, a, b)$. ∎

We have some more definitions to make.

If $M_l[x, y] \leq D(G_h, x, y)$, we say that x and y are *separable*.

Let r be the root of the ultrametric tree U, and a a leaf in U. Define
$\mathcal{L}(U,a) = \{leaves\ s : \mathtt{lca}(a,s) < r\}$

For any two vertices a and b define the graph $G_{a,b}$ as follows:

$G_{a,b} = (V, E_{a,b})$, where $V = \{1,\ldots,n\}$
$E_{a,b} = \{(x,y) | 1 \le x,y \le n \text{ and } M_h[x,y] < M_l[a,b]\}$.

Let $C(G_{a,b},a)$ be the connected component of $G_{a,b}$ containing a and let $C_v(G_{a,b},a)$ be the vertex set of that component.

Lemma 6 *Let $M_l[a,b]$ be maximal. If there exists a tight $U \in [M_l,M_h]$, then there exists a tight $U' \in [M_l,M_h]$ such that $\mathcal{L}(U',a) = C_v(G_{a,b},a)$.*

Proof: We first show that $\forall U \in [M_l,M_h]$, $C_v(G_{a,b},a) \subseteq \mathcal{L}(U,a)$. Suppose that there exists a vertex $x \in C_v(G_{a,b},a) - \mathcal{L}(U,a)$. There must be a path, $a = v_0, v_1, \ldots, v_k = x$, from a to x in $G_{a,b}$. Let $v_i, i > 0$ be the first vertex on the path that is not in $\mathcal{L}(U,a)$. There must exist such a vertex since, by assumption, $v_k = x \notin \mathcal{L}(U,a)$. Since (v_{i-1},v_i) is an edge in $G(a,b)$, $M_h[v_{i-1},v_i] < M_l[a,b]$. On the other hand, $\mathtt{lca}(v_{i-1},v_i) = r$ (the root), so that $d^U_{v_{i-1}v_i} = M_l[a,b]$. But then $d^U_{v_{i-1}v_i} > M_h[v_{i-1},v_i]$, a contradiction.

Now suppose that there exists a tight $U \in [M_l,M_h]$ such that for some non-empty set S such that $S \cap C_v(G_{a,b},a) = \emptyset$, $C_v(G_{a,b},a) \cup S = \mathcal{L}(U,a)$. For any set of leaves V, let $T_U(V)$ be the minimal subtree induced by the leaves in V. We construct U' from U by replacing $T_U(C_v(G_{a,b},a) \cup S)$ with the tree formed by making $T_U(C_v(G_{a,b},a))$ and $T_U(S)$ each a child of a root.

Let x and y be any leaves in U'. If $x,y \notin \mathcal{L}(U,a)$, then $M_l[x,y] \le d^{U'}_{xy} = d^U_{xy} \le M_h[x,y]$. If $x \notin \mathcal{L}(U,a)$ and $y \in \mathcal{L}(U,a)$, then $d^{U'}_{xy} = d^U_{xy} = \mathtt{MAX}[U]$. If $x,y \in S$ or $x,y \in C_v(G_{a,b},a)$, then once again $d^{U'}_{xy} = d^U_{xy}$. Finally, if $x \in S$ and $y \in C_v(G_{a,b},a)$, we have $d^{U'}_{xy} = \mathtt{MAX}[U'] = \mathtt{MAX}[U] = \mathtt{MAX}[M_l]$. But $M_h[x,y] \ge \mathtt{MAX}[M_l]$ since $x \notin C_v(G_{a,b},a)$, so $M_l[x,y] \le d^{U'}_{x,y} \le M_h[x,y]$ and $U' \in [M_l,M_h]$. ∎

The lemma above implies the following algorithm:

Algorithm A *Sandwich to Ultrametric - Greedy*

- Find (a,b) that maximizes $M_l[a,b]$.
- Construct $G_{a,b}$.
- Make each connected component of $G_{a,b}$ a child of the root of U.
- Recurse on each connected component.

Theorem 2 *Algorithm A is correct and runs in time $O(en)$.*

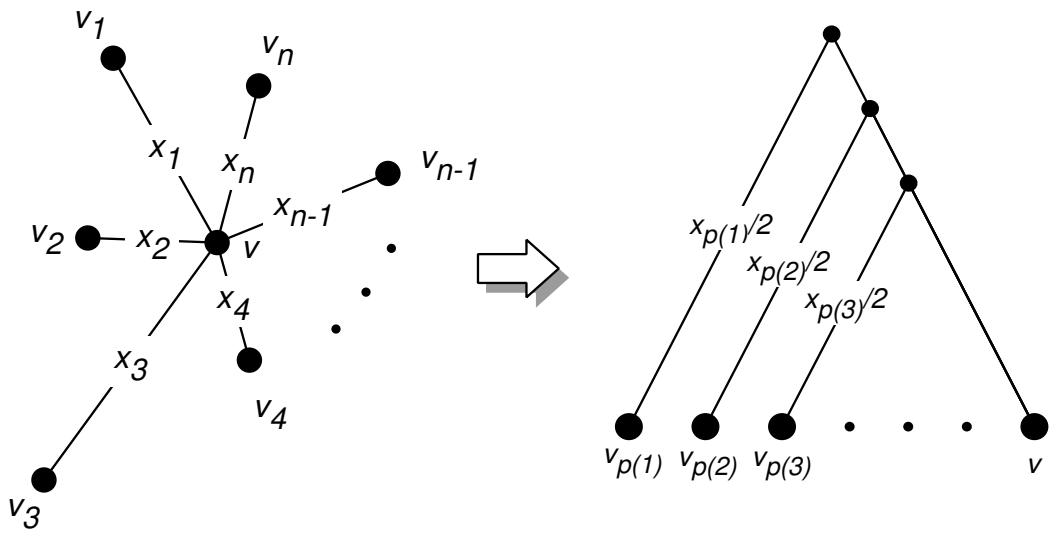

Figure 1: Ultrametric tree satisfying distance constraints imposed by the reduction

reduction is linear time, the $\Omega(n \log n)$ lower bound on sorting implies a similar lower bound on the Graph Sandwich to Ultrametric Problem. The $\Omega(e)$ lower bound on the problem follows from the observation that we need to look at every edge in the input. ∎

4.2 An $O(en)$ algorithm

We first describe a simple $O(en)$ algorithm to motivate the optimal algorithm. Before presenting the algorithm, we introduce some lemmas.

Lemma 4 *If* $\exists U \in [M_l, M_h]$, *then* $\exists U' \in [M_l, M_h]$ *such that* $\texttt{MAX}[U'] = \texttt{MAX}[M_l]$.

Proof: Suppose otherwise that no such U' exists and let $U'' \in [M_l, M_h]$ be a tree that minimizes $s = \texttt{MAX}[U''] - \texttt{MAX}[M_l] > 0$. From amongst the edges incident on the root of U'', let the minimal edge weight be w. Let $\hat{U}$ be the tree derived from U'' be shortening all edges incident on the root by $\min\{w, s/2\}$. We will show that $\hat{U} \in [M_l, M_h]$ thus violating the minimality of U''.

Let a and b be any leaves in U''. Then $M_l[a,b] \leq d^{U''}_{a,b} \leq M_h[a,b]$. In producing $\hat{U}$, we shorten some edges but do not lengthen any edges so we know that $d^{\hat{U}}_{ab} \leq M_h[a,b]$. Now we must show that $M_l[a,b] \leq d^{\hat{U}}_{ab}$. We have two cases. If $\texttt{lca}(a,b)$ is below the root in U'', then $d^{U''}_{ab} = d^{\hat{U}}_{ab}$. Otherwise, $d^{U''}_{ab} = \texttt{MAX}[U''] = \texttt{MAX}[M_l] + s$. But $d^{\hat{U}}_{ab} = d^{U''}_{ab} - 2\min\{w, s/2\} = \texttt{MAX}[M_l] + s - 2\min\{w, s/2\} \geq \texttt{MAX}[M_l] \geq M_l[a,b]$. ∎

If $U' \in [M_l, M_h]$ is such that $MAX[U'] = MAX[M_l]$, then we call U' *tight*. The following corollary is immediate and its proof is omitted.

Corollary 5 *For any tight ultrametric tree* $U \in [M_l, M_h]$, *if* $M_l[x,y] = \texttt{MAX}[M_l]$, *then* $d^U_{x,y} = \texttt{MAX}[M_l]$.

Desired tree	Graph sandwich	Min increment L^1-norm	Min increment L^∞-norm	Min Tree Size
Ultrametric	$\Theta(e + n\log n)$ § 4	NPC § 6	$\Theta(e + n\log n)$ § 4.5	Not-Approx § 8
Additive	NPC § 5.1	NPC § 7		

Recall the definition of the *Sandwich to Ultrametric* problem.

Given: Graphs, G_h and G_l with respective weight functions, M_h and M_l.

Output: An ultrametric tree U (if one exists) such that for all $(i,j) \in E_l$, $M_l[i,j] \leq d^U_{ij}$ and for all $(i,j) \in E_h$, $d^U_{ij} \leq M_h[i,j]$.

Whenever such a tree U exists between G_l and G_h, we will say that $U \in [M_l, M_h]$. Although these weight functions are partial, it will be convenient in the description below to extend them to total functions as follows: Whenever $(i,j) \notin E_l$, set $M_l[i,j] = 0$ and whenever $(i,j) \notin E_h$, set $M_h[i,j] = \infty$. This extension is for ease of description and these additional values will not be looked at by any of our algorithms.

We make some definitions which will be useful in the ensuing lemmas. Given a weight function M, we will let $\texttt{MAX}[M] = \max\{M[i,j]\}_{i,j=1}^n$ Also, we define $\texttt{MAX}[U] = \max\{d^U_{ij}\}_{i,j=1}^n$ for tree U. There is a natural way of rooting ultrametric trees, so that the root is equidistant from every leaf. We can then define the *least common ancestor* of every pair of nodes, x, y. This will be denoted by `lca(x,y)`. Note that this rooting also defines a partial order on the nodes in the tree. We will say that $v \leq w$ for nodes v, w in the tree if the path from v to the root r passes through w.

4.1 Lower Bounds

Theorem 1 *The* Sandwich to Ultrametric *requires* $\Omega(n\log n + e)$ *time.*

Proof: We prove the lower bound by reduction from sorting. Suppose we wish to sort the numbers $x_1, \ldots, x_n$. (To simplify the exposition we will assume that the x_i's are distinct since the lower bound on sorting holds even in this special case.) We create an instance of the Sandwich to Ultrametric problem as follows. Let $G = (V, E, W)$ where $V = \{v\} \cup \{v_1, \ldots, v_n\}$, $E = \{(v, v_i) | 1 \leq i \leq n\}$, and $W(v, v_i) = x_i$. Set $G_l = G_h = G$. Now there is always an edge weighted ultrametric binary tree, U such that $d^U_{(v,v_i)} = x_i$ for all i. Suppose p is the permutation of $[1..n]$ such that $x_{p(1)} > x_{p(2)} > x_{p(3)} \cdots > x_{p(n)}$. Then the ultrametric tree of minimum height is obtained by creating a root r and letting the distance from r to v equal $x_{p(1)}/2$, and hanging off vertices v_i at distance $x_i/2$ from v along the path from v to r. In fact, it is easy to see that in any ultrametric tree satisfying the constraints, whenever $i < j$, $\texttt{lca}(v, v_{p(i)} > \texttt{lca}(v, v_{p(j)})$. Figure 1 illustrates this. Thus, a linear time traversal of the tree will give us the sorted order of the x_i's. Since the entire

Optimization Problems .

Given: An edge weighted graph with weight function M.

Output: An edge-weighted tree T which is optimal under one of the following criteria:

L^∞**-best fit:** T minimizes $\max_{i,j}|d_{ij}^T - M[i,j]|$

L^∞**-min increment:** T such that $d_{ij}^T \geq M[i,j]\ \forall i,j$ and $\max_{i,j}(d_{ij}^T - M[i,j])$ is minimized.

L^1**-best fit:** T minimizes $\sum_{i,j}|d_{ij}^T - M[i,j]|$

L^1**-min increment:** T such that $d_{ij}^T \geq M[i,j]\ \forall i,j$ and $\sum_{i,j}(d_{ij}^T - M[i,j])$ is minimized.

As usual, we distinguish between the case that the desired tree is ultrametric and the case that it is additive by adding the suffix 'to ultrametric' or 'to additive' respectively to the names of each of these problems.

We also consider the problem of incrementing the matrix entries in order to minimize the total size of the resulting ultrametric tree. Here the size of the tree is the sum of the weights of the edges in the tree. We term this problem the *Min. Tree Size* problem.

Several of these optimization problems have been considered in a different context by Křivánek[27]. More specifically [27] is concerned only with ultrametric trees and considers the optimization problems of *best-fit, min increment,* and *min decrement* under the L^∞ norm and gives an $O(n^3)$ algorithm for each of these three problems. In addition [27] considers the problem of minimizing the number of entries to be changed in the input matrix in order for the distances to fit an ultrametric tree when the ultrametric is unrestricted, required to be an increment, and required to be a decrement respectively and shows that the first two of these problems are NP-complete. Although, some of the structural theorems that we prove in this paper are similar to those in [27], we derive significantly more efficient algorithms from these theorems.

Some additional remarks are in order regarding the distinctions between the three problems of best-fit, min increment, and min decrement to an ultrametric. At first glance, the differences between them appear to be merely cosmetic. However, this is not the case when the goal is an ultrametric tree. For example, implicit in [27] is an $O(n^3)$ algorithm for finding the min. decrement to an ultrametric tree under the L^1-norm. In contrast, we show in this paper that the min. increment to an ultrametric under the L^1-norm is *NP-hard.* The best-fit version was also shown *NP-hard* by Day in [9], as noted earlier.

Let $n = |V_l| = |V_h|$, and $e = |E_h \cup E_l|$. Note that when each pair of distances $d(i,j)$ is bounded from above and below, $e = O(n^2)$. We summarize our results along with the section numbers in which they occur in the table below:

4 Results for Ultrametric Trees

In this section we show that the *graph sandwich to ultrametric problem* can be solved in $O(e + n\log n)$ time, and the problem also has a matching lower bound.

on edge $e = (u, v)$ will be denoted $M(e)$ or $M[u, v]$. In all the problems we consider, we will assume that the weights in the input are non-negative, bounded precision real numbers. However this assumption is not crucial and all of our algorithms can handle negative weights after minor modification.

Graph Sandwich Problem

Input: Two edge-weighted graphs $G_h = (V, E_h, M_h)$ and $G_l = (V, E_l, M_l)$.

Output: An edge-weighted tree T (if possible) such that $M_l[i, j] \leq d_{ij}^T$ for all $(i, j) \in E_l$ and $d_{ij}^T \leq M_h[i, j]$ for all $(i, j) \in E_h$.

Since T is a tree, d^T is necessarily an *additive* metric. We may also require that T be ultrametric. We distinguish between these two variations by calling the first the *Graph Sandwich to Additive Problem*, and the second the *Graph Sandwich to Ultrametric Problem.* Clearly, the above problem is a generalization of the following two problems, each of which has two versions depending on whether the tree sought is ultrametric or merely additive.

Matrix Sandwich Problem

Given: Two $n \times n$ matrices, M_l and M_h, where M_l is the *lower bound* and M_h is the *upper bound.*

Output: An edge-weighted tree T (if possible) such that $M_l[i, j] \leq d_{ij}^T \leq M_h[i, j]$ for all i, j.

Matrix Completion Problem .

Given: An $n \times n$ matrix M which is only partially filled in, so that for certain entries i, j we have $M[i, j] =$ "*" (which indicates that $M[i, j]$ is *unset*).

Output: An edge-weighted tree T so that if $M[i, j] \neq$ "*", then $d_{ij}^T = M[i, j]$, if such a tree exists.

We also consider optimization problems where the goal is to arrive at an edge-weighted tree whose interleaf distances most 'closely fit' a given matrix of distances. We consider both the L^1 and L^∞ norms to evaluate closeness of fit. We denote these problems as $L^\infty-$*best fit* and L^1-*best fit.* We consider a variation of the closest fit problems where we are only allowed to increment the entries in the matrix. Since observed distances are frequently lower bounds on the tree-distance, this assumption has practical relevance. For example, this is the case when the distances are derived from alignments of biomolecular sequences for the species. In fact, one of the original and most popular heuristics for constructing trees from distance matrices, the *Distance Wagner Method*[12] proposed by Farris in 1972, restricts attention to trees T with this property. Again under the two norms discussed, we term these problems, $L^\infty-$*min increment* and L^1-*min increment.* Each of the problems described above has a variation where the tree sought is not just additive, but also ultrametric. A formal description of all of these problems is given below.

In [38] Waterman et. al proved that at most one edge-weighted tree can realize an additive distance matrix (assuming that no edge weight is zero) and provided an $O(n^2)$ algorithm to construct such trees from additive distance matrices. Subsequently, $O(n \log n)$ algorithms to construct trees from additive distance matrices for bounded degree trees were found by several authors (see for example [8]).

Ultrametric distance matrices also have two nice alternate characterizations.

Fact 2 For all sets of three leaves $\{i, j, k\}$, $M[i, j] \leq max\{M[i, k], M[j, k]\}$, or equivalently,
Any three points of S can be called i, j, k so that $M[i, j] \leq M[j, k] = M[i, k]$

One other characterization of ultrametric distances arises from thinking of distance matrices as edge-weighted graphs. This obvious correspondence is used frequently in this paper, so we spell it out here. Given an $n \times n$ symmetric matrix M we can represent the information in the matrix by an edge-weighted n-node graph where the weight on the edge (i, j) is equal to the matrix entry $M[i, j]$. In what follows, we will sometimes be concerned with partial matrices where some of the values are unspecified. It is easy to model such a matrix as a graph where we only have the edges that correspond to the specified entries.

Fact 3 *[23] An additive distance matrix M is ultrametric if and only if for every cycle in the weighted graph defined by M, the maximal weight edge is not unique, i.e. the maximal value appears more than once.*

Efficient algorithms for constructing trees from ultrametric distance matrices were found by [25] and others.

Unfortunately, interspecies distances are rarely even additive, so that rather than seeking a tree which exactly fits the distance matrix M, biologists seek to find an edge-weighted tree T which minimizes the following *cost*:

$$C(T, M) = \sum_{i,j} w_{ij} |d_{ij}^T - M[i, j]|^\alpha$$

Here, w_{ij} are weights and and $\alpha > 0$. Various proposals for the w_{ij} were given by several authors [16, 3, 38]. Setting $w_{ij} = 1$ was suggested by Cavalli-Sforza and Edwards in [6], but in [9], Day showed that when $w_{ij} = 1$, finding optimal trees for the above criterion is *NP-hard*, for $\alpha = 1$ or 2, and for T required to be ultrametric or only additive.

3 New Models of Computation

The central problem we will be concerned with is called the *graph sandwich problem* and is a generalization of several reasonable ways of modeling inaccuracies in the distance information. We now describe this problem and the relevant specializations of this problem. We will refer to an edge-weighted graph as a triple, $G = (V, E, M)$, where V is the set of vertices, E is the set of edges, and M is the weight function on the edges. The weight

the Dayhoff matrix [10] is commonly used, but even the values in this matrix are not universally accepted (see for example [2, 18, 39]). Thus, the distance between species i and j computed from an 'optimal' pairwise alignment may not be very reliable. Using multiple alignments only exacerbates many of the problems mentioned above, for in contrast to pairwise alignments for which optimal alignments can be found efficiently, optimal multiple alignments are computationally expensive to find (all known algorithms for finding optimal multiple alignments, under a variety of criteria, use exponential time, and some are known to be *NP-hard*[35]).

2 Preliminary Definitions

We define some simple terms in this section which are in use in the biological literature.

Definition 1 *A* **phylogenetic tree** *for a species set S is a rooted tree in which the leaves are labeled by the species in S, and the internal nodes represent the ancestors of the species.*

Definition 2 *A distance matrix M is said to be* **additive** *if it is possible to construct a tree T together with weights on the edges such that for all i, j, $M[i,j] = d_{ij}^T$.*

Definition 3 *An edge-weighted tree T is called* **ultrametric** *if it can be rooted in such a way that the lengths of all the root-leaf paths in the tree are equal. An additive distance matrix, M, is called* ultrametric *if it represents the inter-leaf distances of an ultrametric tree.*

The primary motivation for using ultrametric distance matrices is that distances (as measured in time) are ultrametric. That is, if we construct an evolutionary tree with the internal nodes indicating speciation, then the natural distance to place on that edge is the amount of time between the speciation events. In this case, if we define the distance function d_{ij} to be equal to twice the time since species i and j diverged from a common ancestor, then $d_{ij} = d_{ij}^T$, so that a matrix M with $M[i,j] = d_{ij}$ is *additive*. If in addition we require that the species in our set S (which will occupy the leaves of the evolutionary tree we construct) are all *extant* species, then the distance from the root to each leaf is the same, so that the tree is *ultrametric.*

A simple characterization of when a distance matrix is additive was discovered in 1971 by Buneman in [5].

Fact 1 *A distance matrix M is additive if and only if given any four points from the set of points on which the distances are defined, we can name them i, j, k, l in such a way that*

$$M[i,j] + M[k,l] = M[i,k] + M[j,l] \geq M[i,l] + M[j,k]$$

This condition is called the *four-point* condition. When the pairwise distances between i, j, k, and l satisfy the constraint above with strict inequality, it is easy to see that in any tree realizing these distances, there is an edge e whose removal causes i and l to lie on one side and j and k to lie on the other.

1 Introduction

An *evolutionary tree* for a species set S is a rooted tree in which the leaves represent the species in S, and the internal nodes represent ancestors. The most frequently used models of computation for constructing these trees use either *characters* or *distance matrices*. Characters are functions from the species sets to integers, and may be based on morphological data (such as the character *invertebrate-vertebrate*, which has two *states*) or molecular data. An example of a character arising from molecular data is the character whose value is the nucleotide present at a specific position in a multiple alignment of DNA sequences for S, in which case there are four possible states – A,C,T, or G. Constructing trees from characters is quite popular, especially recently with the tremendous increase in sequence information. However, the problem of constructing trees from characters in general is that constructing optimal trees from such data is *NP-hard* for most interesting cases. Biologists interested in computing trees from characters have therefore resorted to heuristics which find trees which are *locally* optimal, so that no local rearrangement improves the score for the tree.

The other standard way of computing trees uses distances. Here, the input to the problem is an $n \times n$ distance matrix M where $M[i,j]$ is the observed distance between species i and j. Given such a matrix, the objective is to find an edge-weighted tree T in which the distance d_{ij}^T in the tree between the leaves for i and j (defined to be the sum of the weights in the path between i and j) is such that $C(T,d) = \sum_{i,j} |d_{ij}^T - M[i,j]|^\alpha$ is minimized, where various choices of α correspond to various norms. When it is possible to define the edge-weighted tree T so that $C(T,d) = 0$, then the distance matrix is said to be *additive*. A polynomial time algorithm for reconstructing trees from additive distances was given by Waterman et. al [38], who proved in addition that at most one tree can exist. Their algorithm uses $O(n^2)$ time. However, distances drawn from alignments of molecular sequences are hardly ever additive (see [38] for a nice explanation of why this is), and in [9], Day showed that the optimization problem is *NP-hard*. Despite these difficulties, biologists still compute trees from distances, and currently a multitude of methods exist for this purpose. Unfortunately, these methods do not find optimal trees in all cases, and either amount to unproven heuristics ([31, 33, 37, 12, 28] and others) or exhaustive search.

We motivate our models further with a brief discussion of why it is hard to obtain accurate interspecies distances. There are several ways of computing these distances. Currently the most common methods use alignments of biomolecular sequences for the species. Constructing distances based upon alignments assumes that a biomolecular (i.e. DNA, RNA, or amino-acid) sequence is obtained for each species in S, and the sequences are *homologous*, so that they correspond for example to genes with the same function. There are then two possibilities: the distance between species i and j is the score of the optimal pairwise alignment between the associated sequences, or it can be the score of the pairwise alignment of the sequences for species i and j induced by an optimal *multiple* alignment. For pairwise alignments, there is no general agreement about how to determine the *correct* pairwise alignment, whether a given alignment is correct, and how to score a given alignment. Issues such as what the relative costs are of various edit operations, how to score a gap of length k, etc., are the source of much debate in the computational biology community [30, 38]. When aligning amino acid sequences,

ABSTRACT

Constructing evolutionary trees for species sets is a fundamental problem in computational biology. One of the standard models assumes the ability to compute distances between every pair of species and seeks to find an edge-weighted tree T in which the distance d_{ij}^T in the tree between the leaves of T corresponding to the species i and j exactly equals the observed distance, d_{ij}. When such a tree exists, this is expressed in the biological literature by saying that the distance function or matrix is *additive*, and trees can be constructed from additive distance matrices in $O(n^2)$ time. Real distance data is hardly ever additive, and we therefore need ways of modeling the problem of finding the best-fit tree as an optimization problem.

In this paper we present several natural and realistic ways of modeling the inaccuracies in the distance data. In one model we assume that we have upper and lower bounds for the distances between pairs of species and try to find an additive distance matrix between these bounds. In a second model we are given a partial matrix and asked to find if we can fill in the unspecified entries in order to make the entire matrix additive. For both of these models we also consider a more restrictive problem of finding a matrix that fits a tree which is not only additive but also *ultrametric*. Ultrametric matrices correspond to trees which can be rooted so that the distance from the root to any leaf is the same. Ultrametric matrices are desirable in biology since the edge weights then indicate evolutionary time. We give polynomial time algorithms for some of the problems while showing others to be NP-complete. We also consider various ways of 'fitting' a given distance matrix (or a pair of upper and lower bound matrices) to a tree in order to minimize various criteria of error in the fit. For most criteria this optimization problem turns out to be NP-hard, while we do get polynomial time algorithms for some.

DIMACS Technical Report 93-22
April 1993

A Robust Model for Finding Optimal Evolutionary Trees

by

Martin Farach[1] Sampath Kannan[,2] Tandy Warnow[,3]
DIMACS U. of Arizona Sandia National Labs

[1]DIMACS, Box 1179, Rutgers University, Piscataway, NJ 08855; (908) 932-5928; farach@dimacs.rutgers.edu; Supported by DIMACS under NSF contract STC-88-09648.
[2]Department of Computer Science, University of Arizona, Tucson, AZ 85721; (602) 621-4817; kannan@cs.arizona.edu; Supported by NSF Grant CCR-9108969.
[3]Algorithms and Discrete Mathematics Department, Sandia National Labs. Albuquerque, NM; (505)-845-7604; twarnow@cs.sandia.gov. This work was begun while this author was visiting DIMACS in July and August, 1992, and was supported in part by the U.S. Department of Energy under contract DE-AC04-76DP00789

DIMACS is a cooperative project of Rutgers University, Princeton University, AT&T Bell Laboratories and Bellcore.

DIMACS is an NSF Science and Technology Center, funded under contract STC–91–19999; and also receives support from the New Jersey Commission on Science and Technology.